WebObjects Performance Optimization

Session 637

These are confidential sessions—please refrain from filming or taking pictures.

Enterprise IT

WebObjects Performance Optimization

Bill Bumgarner, Max Muller, and Ravi Mendis

http://developer.apple.com/wwdc2005/

Defining the Performance Target

The landscape has changed

- Bandwidth (and lots of it)
- Current “state of the art” HTML UIs
- Alternative Interface technology
- How has the task of optimizing WebObjects solutions changed?

Defining the Target Market

Bandwidth and lots of it

- Broadband is both much more common and commonly a requirement
- More emphasis on multimedia content
  - Therefore, the piddly little bit o' HTML is often noise in comparison
- Server environments also have more bandwidth
  - Servers have faster connections to the net at large
  - Server room bandwidth is cheaper
  - Interconnects between machines are faster now, too

Defining the Target Market

Current "state of the art" HTML UIs

- *Almost nothing has really changed in the last few years!*
  - CSS shuffled the bits around, but didn't really change the overall architecture
- The baseline feature set to create a "real" site have changed significantly
  - More inferential content
  - More per-user content
    - Customizations
    - Marketing and related items
  - Increased integration with external resources

Defining the Target Market

Alternative Interface technology

- It isn't just about HTML anymore
- "Web Services"—an over-hyped word that means "XML interface"
  - RSS is an XML view into a site's data
  - iTunes Music Store uses XML on the wire for the UI
  - Many Dashboard widgets use web services to glean information
- HTML is also being used in new contexts
  - Xcode's documentation engine can display content from ADC's servers
  - WebKit allows HTML views to be embedded anywhere
- App servers frequently have to talk with other app servers
- End result: Your application will be generating content destined for use in context far outside of a traditional browser

Defining the Target Market

How has the task of optimizing WebObjects solutions changed?

- All about the database, active monitoring and adaptive load balancing
- Increased use of "inferential" content means content must often be dynamically generated when static would do before
  - But "dynamic" does not mean "real time"!
- Higher complexity applications or apps with large traffic loads often exhibit failure modes that *can only be analyzed in a production environment!*
  - Result: Instrumentation within the production environment is critical!
- Applications can and do go insane. Traffic peaks happen. Usage patterns do change
  - Adapting to changing requirements increases usability and can decrease deployment costs

Design Optimization

http://developer.apple.com/wwdc2005/

“True Refinement Seeks Simplicity”

Bruce Lee

Design Goal #1: Keep It Simple

Re-Engineering is Expensive!

- Model is hard to re-engineer
  - Data is hard to migrate in production
- Get it right from the start!

Keep the Model simple

- Don’t over-engineer
  - Concrete > Abstract

Minimize inheritance

- Use Single table inheritance

Modeling Inheritance

EOF Inheritance Comparison Chart

Inheritance	Fetches
Single-Table	1
Horizontal	n
Vertical	n X m

Modeling Inheritance

Flat inheritance trees

- Avoid deep hierarchies
  - Ideally 1 level deep
- Always use abstract super-classes

Do not combine inheritance methodologies

- No mix and match

Be economical with inheritance

iTunes Music Store

- 400+ entities
- A dozen models
- 1 inheritance hierarchy 1-level deep
- 5% of entities use inheritance

Design Goal #2: Exploit the DB

Sub-queries

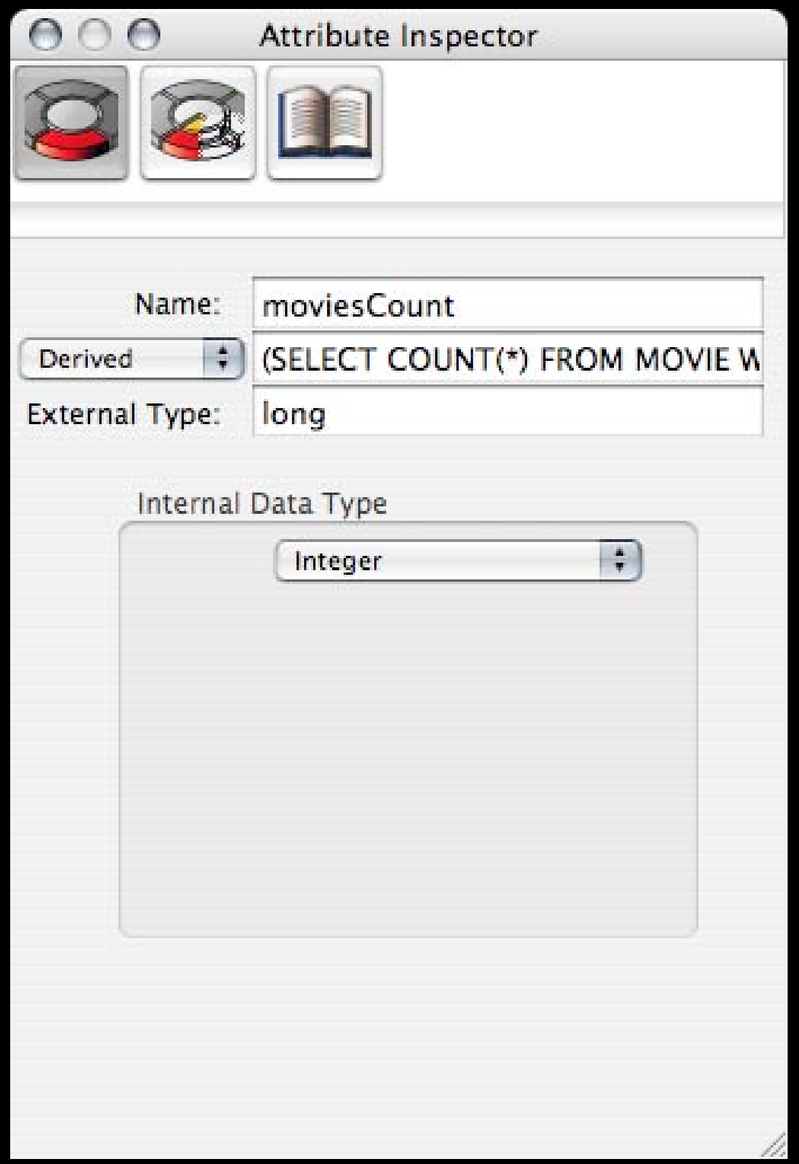
- Use for Aggregation

Raw Rows

- Use for fast search results

Pre-fetching

- Use for detailed inspection/edit page



Sub-queries

How to

- Derived attribute
- SQL for Sub-query in column
- Set as read-only, no locking

Example

- Studio moviesCount
- (SELECT COUNT(*) FROM MOVIE WHERE MOVIE.STUDIO_ID = STUDIO.STUDIO_ID)

Sub-queries

Advantages

- Aggregate SQL functions: COUNT(), MAX(), MIN(), etc.
- Fast
- Can be implemented as EO attributes
  - No raw SQL

Caveats

- Potential performance penalty with large data sets

Case Study

- iTunes Music Store

Raw Rows

Advantages

- Lightweight NSDictionaries as raw rows
- Optimal SQL SELECT/JOIN

Disadvantages

- Faulting required to convert raw rows to EOs

When to use

- Fast searches
- List pages
- Cache (read-only) non-reference data

A3XX
A3XX

Design Goal #3: Minimize Memory Footprint

Optimize session state management

- Use DirectActions and stateless components

Share data across sessions

- Leverage EOSharedEditingContext

Use standard APIs

- WebObjects
- Java
  - log4j

Design Goal #3: Minimize Memory Footprint

Partition application functionality

- Case Study: iTunes Music Store
  - (Pseudo) Web Service front end
  - Content Management Tool

Optimize Session State Management

Maximize use of Stateless pages

- Use DirectAction pages where ever possible
- Fetch EOs into local editing contexts
  - Results discarded at the end of the request-response loop
- Implement stateless WebObjects components
  - Singleton

Minimize Session State

- Use Session vars sparingly

EOSharedEditingContext

Advantages

- Shares read-only data between sessions
- Thread safe
- Reduces DB traffic
  - Avoids frequent and repeat fetches

Case Study

- iTunes Music Store
  - 40+ shared entities
  - Significant performance boost

Optimization Techniques

Key = Observation

- Identify bottlenecks
- Monitor usage patterns

Analysis

- Set EOAdaptorDebugEnabled = YES
- Profile with WOEvents page
- Monitor usage with WOStats page
- Use web server log analysis tools
  - WOStatisticsStore logString()

Tuning and Diagnostics

http://developer.apple.com/wwdc2005/

WOAdaptor Tuning

Default settings not always what you want

- Connect/Send/Recieve Timeouts
- Retries
- Dead Timeout
- What “No Instance Available” really means

MZStore L/B: roundrobin redir: /StoreNotAvailable.xml dead time: 20 max pool sz: 0 retries: 3	**inst**	**host**	**port**	**active reqs**	**served**	**conn pool peak/reused**	**cto / sto / rto**	**send/rcv buf**	**refusing timeout**	**dead timeout**
	1	mzstore001	12001	0	515	0/0	5/20/30	32768/32768	0	0
	2	mzstore002	12002	0	496	0/0	5/20/30	32768/32768	0	0
	3	mzstore003	12003	0	502	0/0	5/20/30	32768/32768	0	0

WOApp Server Tuning

- WOListenQueue Size—the number of requests queued to hand off to worker threads. Defaults to 128!
- WOWorkerThread Size—the number of active threads that can be processing requests or waiting for the request handler lock. Defaults to max of 256!
- Words of caution: Killing long running threads that are performing EOF operations can leave the entire stack horked.

Deployment Tuning

- Guard against one application/set of applications taking out your web servers
- Heterogeneous mixture of applications among servers
- Java Max VM size != Memory footprint of Java application. As a rule of thumb add 100 MB per Java instance.
- Time to add more instances when average idle time < 0.5 seconds, CPU load is sustained above # of CPUs or system reacts slowly/flat lines under load
- Always bounce applications in a rolling fashion

iTMS Diagnostic Aids

- Stats collections with standard deviations
- Watchdog thread that monitors long running requests
- Detection of large DB fetches
- Statistical graphs over time
- EditingContext stats (time created, registered objects)
- SQL stats (most frequent queries, recent queries)
- Total properties view as well as ability to reload on the fly
- Independent web servers for gathering stats and monitoring

Demo

http://developer.apple.com/wwdc2005/

More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Who to Contact

Alan Samuel
Technology Evangelist
bluker1@apple.com

WebObjects Feedback
webobjects@apple.com

Q&A

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference